A Comparative Study of Learning-to-Rank Techniques for Tag Recommendation

Sérgio D. Canuto, Fabiano M. Belém, Jussara M. Almeida, Marcos A. Gonçalves

Computer Science Department
Universidade Federal de Minas Gerais, Brazil
{sergiodaniel, fmuniz, jussara, mgoncalv}@dcc.ufmg.br

Abstract.

Tags have become very popular on the Web 2.0 as they facilitate and encourage users to create and share their own content. In this context, there is a large interest in developing strategies to recommend relevant and useful tags for a target object, improving the quality of the generated tags and of the Information Retrieval (IR) services that use them as data source. Several existing tag recommendation strategies treat the problem as a multiple candidate tag ranking problem, recommending tags that are in top positions of the generated ranking. This motivates the use of Learning-to-Rank (L2R) based strategies to automatically "learn" good tag ranking functions. However, previous work has explored only three different L2R techniques, namely, *Genetic Programming* (*GP*), *RankSVM* and *RankBoost*, comparing at most two of them with respect to effectiveness. In contrast, we here perform a much more comprehensive comparative study of the use of L2R techniques for tag recommendation. Specifically, we compare eight different L2R techniques, namely, *Random Forest* (*RF*), *MART*, λ-*MART*, *ListNet*, *AdaRank* and the three aforementioned techniques, with respect to both effectiveness (i.e., precision, NDCG) and efficiency (i.e., time complexity). We perform experiments using real data collected from five popular Web 2.0 applications, namely, Bibsonomy, LastFM, MovieLens, YahooVideo and YouTube. Our results show that the best L2R based strategy significantly outperforms the best state-of-the-art unsupervised technique (by up to 29% in NDCG). Moreover, unlike existing comparisons of different L2R techniques in other domains, we find that, for tag recommendation, there is a clear winning group of methods (*RF*, *MART* and λ-*MART*) with a slight advantage of two (RF and λ-*MART*) over the other, with gains in NDCG ranging from 4% to 12% over the best of the remaining alternatives considered. We also find that recommendation time, despite some variation among the different methods, is under 1.3 seconds, on average (in the worst case scenario), for all L2R methods, which confirms the feasibility of the L2R approach for tag recommendation.

Categories and Subject Descriptors: H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.5 [**Information Storage and Retrieval**]: Online Information Services

Keywords: learning-to-rank, relevance metrics, tag recommendation

1. INTRODUCTION

The Web 2.0 is characterized by the central role played by users in the creation and sharing of their own content. This content usually comprises a main *object* (e.g., a video) and several sources of data associated with it, referred to as its *features*. An object's *textual features* are well defined blocks of text such as title, tags, description and user comments. Among all textual features, tags have attracted special attention as they offer an effective data source for Information Retrieval (IR) services such as search and classification [Figueiredo et al. 2013]. Thus, there is a large interest in developing strategies to recommend tags to users, improving the quality of the generated tags and, indirectly, of the IR services that rely on them as data source.

This research is partially funded by the Brazilian National Institute of Science and Technology for Web Research (MCT/CNPq/INCT Web Grant Number 573871/2008-6), and by the authors' individual grants from CNPq, CAPES and FAPEMIG.

Most existing tag recommendation strategies treat the problem as a multiple candidate tag ranking problem, sorting tag candidates according to some metric of relevance and recommending tags that are in the top positions of the generated ranking [Belém et al. 2011; Lipczak and Milios 2011; Wu et al. 2009]. The relevance of a candidate usually refers to how well it *describes* the content of the target object or helps to *discriminate* it from other objects. This modeling of the problem motivates the use of Learning-to-Rank (L2R) based strategies to automatically "learn" good tag ranking functions. However, previous work on tag recommendation has explored only three different L2R techniques, namely, Genetic Programming (GP), RankSVM and RankBoost. While Cao et al. [2009] proposed the use of RankSVM, Wu et al. [2009] explored RankBoost, although neither compared the adopted approach with alternative L2R methods. Similarly, in previous work [Belém et al. 2011], we investigated the use of both GP and RankSVM to design tag recommendation methods, comparing them against each other as well as novel unsupervised heuristic methods that extended previously proposed strategies. Our results indicated that L2R techniques provide some gains over the best heuristic. Moreover, the flexibility of the L2R framework in terms of the incorporation of new attributes and ability to maximize different target measures makes it an attractive solution for the tag recommendation problem.

These previous studies of L2R techniques for tag recommendation are somewhat limited since: (1) only three techniques have been considered, (2) only two techniques have been directly compared against each other, and (3) the considered techniques have been evaluated with respect to effectiveness only (efficiency aspects have been disregarded). However, given the success of L2R on several other domains [Gomes et al. 2013; Faria et al. 2010], there is currently a large body of work on L2R techniques. Moreover, the efficiency of these techniques in terms of recommendation time, which is a time sensitive online task, should be taken into account when comparing them. This is an important performance measure since the user must wait for recommendations to be provided and may abandon the system if the waiting time is beyond a certain level.

In this context, we here perform an extensive evaluation of the use of different L2R techniques for tag recommendation. Our main contributions are threefold. First, we compare eight L2R techniques, including the three aforementioned methods as well as five techniques that have not been previously exploited for tag recommendation. These techniques are *Random Forest* (RF), $MART$, λ-$MART$, $ListNet$ and $AdaRank$. Second, our evaluation comprises not only effectiveness (e.g., precision, NDCG [Baeza-Yates and Ribeiro-Neto 1999] and recall of recommended tags), but also efficiency (i.e., recommendation time), which has been ignored in previous work. Third, our experiments exploit real data collected from five popular Web 2.0 applications, namely, Bibsonomy, LastFM, MovieLens, YahooVideo and YouTube[1], which is a much larger number of datasets than used by previous work. We compare all eight L2R methods against each other as well as against a state-of-the-art heuristic (the best unsupervised method proposed by Belém et al. [2011]). To our knowledge, this is the most comprehensive experimental evaluation of the use of L2R methods for tag recommendation.

Our results show that, unlike existing comparisons of different L2R techniques in other domains (e.g., document ranking) [Gomes et al. 2013], for tag recommendation, there is a clear winning group of methods (RF, $MART$ and λ-$MART$), with a slight advantage of two (RF and λ-$MART$) over the other, with gains in NDCG [Baeza-Yates and Ribeiro-Neto 1999] ranging from 4% to 12% over the best of the remaining L2R techniques considered. We also find that recommendation time, despite some variation across methods, is under 1.3 seconds, on average, for all L2R techniques, in a worst case scenario in which no pre-computed data is available in cache. Thus, this recommendation time is reasonably short for an interactive task. Finally, we find that the best L2R method significantly outperforms the best unsupervised solution, with gains in NDCG of 29%, on average. These results confirm the feasibility and effectiveness of the L2R approach for the tag recommendation task.

The rest of this article is organized as follows. Section 2 discusses related work, whereas Section

[1] `http://www.bibsonomy.org`, `http://www.last.fm`, `http://www.movielens.org`, `http://www.yahoo.com/video`, and `http://www.youtube.com`, respectively.

3 defines the problem. Section 4 introduces the methods analyzed here, while our experimental methodology and results are presented in Sections 5 and 6, respectively. Finally, conclusions and future work are offered in Section 7.

2. RELATED WORK

One of the main sources of information exploited by tag recommendation methods are co-occurrence patterns computed over a history of tag assignments. In particular, co-occurrences are used to expand an initial set of tags associated with the object that is target of the recommendation [Belém et al. 2011; Wu et al. 2009; Sigurbjornsson and van Zwol 2008; Menezes et al. 2010]. To that end, existing solutions often exploit association rules, i.e., implications of the form $X \rightarrow y$, where the antecedent X is a set of tags, and y is a candidate tag for recommendation, restricting the rules by a confidence threshold. Such methods have been called as *associative tag recommenders*. One example is LATRE - Lazy Associative Tag Recommendation [Menezes et al. 2010], which computes association rules in an on-demand manner, allowing an efficient generation of more complex and potentially better rules.

A few other efforts do not exploit tags previously assigned to the target object, focusing, rather, on other textual features [Lipczak and Milios 2011; Wang et al. 2009]. For example, Lipczak and Milios [2011] extract terms from other textual features (e.g., title) of the target object, expanding them with association rules, and sorting the extracted terms by their usage as tags in a training set. Wang et al. [2009], in turn, use the traditional $TF \times IDF$ metric to extract and rank the most important terms from the object's textual content.

In Belém et al. [2011], we extend LATRE and the best method proposed by Sigurbjornsson and van Zwol [2008] to exploit not only co-occurrence patterns among tags previously assigned to the target object, but also the contents of multiple textual features associated with it, and various metrics of relevance. Our comparison of the proposed strategies against various state-of-the-art baselines, including the original associative methods and a method that exploits only other textual features Lipczak and Milios [2011] showed that our solutions outperform the previous techniques.

In addition to tag co-occurrence and textual features, other dimensions, such as image content [Wu et al. 2009], video content [Pedro et al. 2011] and graph clustering [Song et al. 2011], have also been exploited for tag recommendation. A recent work [Yin et al. 2013] addresses not only the problem of recommending tags, but also of predicting different kinds of relationships (such as relations between users and comments), exploiting a generalized latent factor model and Bayesian inference. The authors found that connecting comments and tags within the same model allows mutual reinforcement and improves predictive accuracy. However, it is not possible to compare this method with our strategies because comments are absent in our datasets. Moreover, even if present in a collection, they are usually noisy and tend to be absent in a large fraction of the objects [Figueiredo et al. 2013]. In another direction, in our work by Belém et al. [2013], we extend our previous methods [Belém et al. 2011] to consider not only relevance, but also novelty and diversity as important aspects in recommender systems. Similarly, other studies tackle the problem of producing *personalized recommendations*, often exploiting the history of tag assignments of all users of the application or of a specific user [Rendle and Schmidt-Thie 2010; Lipczak and Milios 2011]. Our current focus is on recommending tags that are relevant to a target object. We plan to address personalized recommendations in the future.

We are aware of only a few prior efforts to apply learning-to-rank (L2R) techniques to recommend tags. One such effort is our prior investigation [Belém et al. 2011], where both RankSVM and Genetic Programming were used to recommend tags, producing some gains over state-of-the-art (unsupervised) methods. Although such gains are somewhat limited, the flexibility and the easiness at which the solutions can be extended, for example to include as attributes metrics that capture new aspects of the problem (e.g., personalization, novelty), makes L2R a very attractive technique. Two other related efforts are those by Cao et al. [2009] and Wu et al. [2009], which exploit RankSVM and Rank-

Boost as L2R techniques, respectively, but do not compare them against alternative L2R methods. Moreover, Wu et al. [2009] consider only metrics related to tag co-occurrences and image content, and do not exploit textual features, whereas Cao et al. [2009] consider only metrics related to tag frequency, disregarding tag co-occurrence metrics. Instead, we here exploit a much richer set of metrics. Furthermore, these prior efforts focus only on the effectiveness of the L2R based tag recommenders. We here analyze not only their effectiveness but also their efficiency with respect to the time required to produce a recommendation for a target object.

The effectiveness of L2R has been studied in other domains, particularly document and image retrieval [Gomes et al. 2013; Faria et al. 2010]. Gomes et al. [2013] compare 13 L2R techniques for document ranking using various datasets of the LETOR benchmark. They found that: (i) in most datasets the results of all analyzed methods are statistically tied with each other and with an unsupervised strategy that uses the best individual attribute in isolation, and (ii) for the other datasets, there is not a clear winner method. In contrast, in the context of image ranking, Faria et al. [2010] show that all studied L2R methods outperform the unsupervised technique (i.e., ranking by attribute values in isolation), and that two L2R methods, one based on association rules and the other based on Genetic Programming, are clear winners. To the best of our knowledge, no previous work has studied issues related to the effectiveness *and* the efficiency of L2R methods for tag recommendation.

3. TAG RECOMMENDATION: PROBLEM STATEMENT

A Web 2.0 *object* refers to an instance of a media (text, audio, video, image) in an application. There are various features associated with an object. *Textual features*, the main source of information exploited by the tag recommendation strategies considered here, are self-contained blocks of text, usually with a well defined functionality [Figueiredo et al. 2013]. We here exploit the following textual features, commonly found in various applications: *tags*, *title* and *description*.

As in our previous work [Belém et al. 2011], we define the tag recommendation problem as follows. Given a target object o, a set of input tags I_o associated with it, and a set of (other) textual features $F_o = \{F_o^1, F_o^2, ..., F_o^n\}$, where each element F_o^i is the set of terms in textual feature i of object o, generate a set of candidates C_o ($C_o \cap I_o = \emptyset$), sorted according to their relevance to o[2]. Considering this scenario, out of the various tag recommendation methods available, those that have consistently produced the most competitive results often exploit term co-occurrence patterns with tags previously assigned to the target object (i.e., tags in I_o) and possibly other metrics of tag relevance [Belém et al. 2011; Menezes et al. 2010; Lipczak and Milios 2011]. We here include metrics that capture such co-occurrence patterns as attributes exploited by the considered L2R methods (see Section 4.1).

In order to learn such co-occurrence patterns and to compute relevance metrics, we exploit a training set $\mathcal{D} = \{\langle I_d, F_d \rangle\}$, where I_d ($I_d \neq \emptyset$) contains all tags assigned to object d, and F_d contains the term sets of the other textual features associated with d. There is also a test set $\mathcal{O}$, which is a collection of tuples $\{\langle I_o, F_o, Y_o \rangle\}$, where both I_o and Y_o are sets of tags associated with object o. However, while tags in I_o are known (and given as input to the recommender), tags in Y_o are assumed to be unknown and are taken as the relevant recommendations to the target object o (i.e., the *gold standard*). This split of the tags of each test object is done to enable an automatic assessment of the recommendations, as performed in various previous studies [Rendle and Schmidt-Thie 2010; Yin et al. 2013] and further discussed in Section 5.2. Similarly, there is a validation set $\mathcal{V}$ used for tuning parameters and "learning" recommendation functions (see Section 5.2). Thus, each object v in $\mathcal{V}$ also has its tag set split into input tags (I_v) and gold standard (Y_v).

[2]We focus on the scenario when there are some initial tags in the target object (i.e., $I_o \neq \emptyset$), and we want to recommend new (different) tags to it. Our methods are also able to recommend relevant tags to an object with no tags by exploiting other textual features and metrics of relevance. We plan to investigate this scenario in future work.

4. TAG RECOMMENDATION STRATEGIES

In this section, we present the tag recommendation strategies analyzed in this work. We start by introducing in Section 4.1 several metrics used to assess the relevance of candidate tags. Next, in Section 4.2, we present a state-of-the-art unsupervised method, considered here a baseline for comparison, which uses a simple linear combination of two relevance metrics as tag ranking function. In Section 4.3, we present the L2R techniques considered here, and describe how we apply them to automatically "learn" good ranking functions by combining a larger number of relevance metrics.

4.1 Relevance Metrics for Tag Recommendation

We here briefly present several metrics that can be used to estimate the relevance of a candidate term c for tag recommendation. Most of them have been previously applied for recommending tags [Belém et al. 2011; Sigurbjornsson and van Zwol 2008; Menezes et al. 2010; Heymann et al. 2008], and are used here as attributes by the L2R based methods. Each metric fall into one of these categories:

—*Tag Co-occurrence Metrics*: estimates how relevant a candidate c is given a set of input tags that often co-occur with t in the dataset.

—*Descriptive Power Metrics*: estimate how accurately candidate c describes the object's content, which is important for information services that exploit object's semantics.

—*Discriminative Power Metrics*: estimate the capability of c to distinguish the target object from others, which supports tasks such as separating the objects into semantic classes or into levels of relevance regarding a query.

—*Term Predictability*: indicates the likelihood that c occurs as a tag.

Table I summarizes all considered metrics. More details can be found in the work by Belém et al. [2011] and Lipczak and Milios [2011].

4.2 Unsupervised Strategy

We consider as a baseline $LATRE{+}wTS$, a state-of-the-art unsupervised tag recommender that is the best heuristic approach proposed by Belém et al. [2011]. $LATRE{+}wTS$ generates candidate tags for a target object o exploiting: (1) association rules (tags which co-occur with the tags already available in o), and (2) terms extracted from other textual features. The scores assigned to these candidates are a linear combination of the values of metrics Sum and wTS (Section 4.1):

$$LATRE + wTS(c, o, \ell, \alpha) = \alpha Sum(c, I_o, \ell) + (1 - \alpha) wTS(c, o) \tag{13}$$

Parameter α ($0 \leq \alpha \leq 1$) is used as a weighting factor. Note that Sum is computed only over candidates generated from the association rules, whereas wTS is computed for terms extracted from other textual features of target object o.

4.3 Learning-to-Rank Based Strategies

Our goal is to investigate the benefits of applying L2R techniques to the tag recommendation problem. The basic idea is to use such algorithms to "learn" a *good ranking function* based on a list L_m of relevance metrics. We here evaluate eight different techniques, namely, *Genetic Programming* (GP), $RankSVM$, $RankBoost$, *Random Forest* (RF), $MART$, λ-$MART$, $ListNet$, $AdaRank$. Out of these, only the first three have been previously applied to the tag recommendation problem [Cao et al. 2009; Wu et al. 2009; Belém et al. 2011], whereas only the first two have been compared against each other [Belém et al. 2011]. Thus, the application of the other five techniques to tag recommendation as well as a thorough evaluation of all solutions are key contributions of this present work. For all strategies, L_m includes all metrics defined in Table I (Section 4.1), which is a richer set of metrics

Table I. Metrics used to estimate the relevance of a candidate tag c as a recommendation for an object o [Belém et al. 2011; Lipczak and Milios 2011]

	Name	Equation/Description
Tag Co-occurrence	*Sum*	Let X be a set of tags and c a candidate tag. $X \rightarrow c$ is an association rule and $\theta(X \rightarrow c)$ is its *confidence*. *Sum* is defined as: $$Sum(c, I_o, \ell) = \sum_{X \subseteq I_o} \theta(X \rightarrow c), \quad (X \rightarrow c) \in \mathcal{R}, |X| \leq \ell, \tag{1}$$ where $\mathcal{R}$ is a set of association rules computed offline over the training set $\mathcal{D}$, and ℓ is the size limit for the antecedent X. As in our previous work by Belém et al. [2011], we use the $LATRE$ algorithm to generate these rules.
	Sum^+	$$Sum^+(c, I_o, k_x, k_c, k_r) = \sum_{x \in I_o} \theta(x \rightarrow c) \times Stab(x, k_x) \times Stab(c, k_c) \times Rank(c, x, k_r), \tag{2}$$ where $Stab(x, k_x)$ is defined in Eq. (10), and k_x, k_c and k_r are tuning parameters. $Rank(c, x, k_r)$ is equal to $k_r/(k_r + p(c, x))$, where $p(c, x)$ is the position of c in the ranking of candidates according to the confidence of the corresponding association rule (whose antecedent is x).
	$Vote$	$$Vote(c, I_o) = \sum_{x \in I_o} j, \text{ where } j = \begin{cases} 1 & \text{if } (x \rightarrow c) \in \mathcal{R} \\ 0 & otherwise \end{cases} \tag{3}$$
	$Vote^+$	$$Vote^+(c, I_o, k_x, k_c, k_r) = \sum_{x \in I_o} j \times Stab(x, k_x) \times Stab(c, k_c) \times Rank(c, x, k_r),$$ $$\text{where } j = \begin{cases} 1, & if \quad x \rightarrow c \in \mathcal{R} \\ 0, & otherwise \end{cases} \tag{4}$$
Descriptive Power	*Term Spread (TS)*	$$TS(c, o) = \sum_{F_o^i \in F_o} j, \text{ where } j = \begin{cases} 1 & \text{if } c \in F_o^i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$
	Term Frequency (TF)	$$TF(c, o) = \sum_{F_o^i \in F_o} tf(c, F_o^i), \tag{6}$$ where $tf(c, F_o^i)$ is the number of occurrences of c in textual feature F_o^i of object o.
	Weighted Term Spread (wTS)	Let the *Feature Instance Spread* of a feature F_o^i associated with object o, $FIS(F_o^i)$, be the average TS over all terms in F_o^i. We define the *Average Feature Spread* $AFS(F^i)$ as the average $FIS(F_o^i)$ over all instances of F^i associated with objects in the training set $\mathcal{D}$. The wTS is defined as: $$wTS(c, o) = \sum_{F_o^i \in F_o} j, \text{ where } j = \begin{cases} AFS(F^i) & \text{if } c \in F_o^i \\ 0 & \text{otherwise} \end{cases} \tag{7}$$
	Weighted Term Frequency (wTF)	$$wTF(c, o) = \sum_{F_o^i \in F_o} tf(c, F_o^i) \times AFS(F^i) \tag{8}$$
Discriminative Power	*Inverse Feature Frequency (IFF)*	$$IFF(c) = log \frac{|\mathcal{D}| + 1}{f_c^{tag} + 1}, \tag{9}$$ where f_c^{tag} is the number of objects in the training set $\mathcal{D}$ that contain c *associated as a tag*.
	Stability (Stab)	$$Stab(c, k_s) = \frac{k_s}{k_s + |k_s - log(f_c^{tag})|}, \tag{10}$$ where the tuning parameter k_s represents the "ideal frequency" of a term in the data collection.
Term Predictability	*Entropy* (H^{tags})	$$H^{tags}(c) = - \sum_{(c \rightarrow i) \in \mathcal{R}} \theta(c \rightarrow i) log \theta(c \rightarrow i) \tag{11}$$
	Predictability (Pred)	$$Pred(c) = \frac{f_c^{tag,F}}{f_c^F}, \tag{12}$$ where $f_c^{tag,F}$ is the number of objects in the training set $\mathcal{D}$ in which c appears both as a tag and as a term in *any* other textual feature, and f_c^F is the number of objects in which c is a term associated with any of its textual features (except tags).

compared to those exploited in previous work [Cao et al. 2009; Wu et al. 2009]. Particularly, metric

Sum is used with values ℓ=1 and ℓ=3, generating thus two attributes: $Sum(c, o, 1)$ and $Sum(c, o, 3)$.

For all analyzed L2R strategies, the set of candidate tags C_o for each object o is the same produced by our unsupervised strategy $LATRE{+}wTS$, including all terms generated by association rules and all terms extracted from other textual features. For each candidate $c \in C_o$ for object o, we compute all metrics in L_m using the training set $\mathcal{D}$ and the other textual features associated with o. Each candidate c is then represented by a vector of metric values $M_c \in \mathbb{R}^a$, where a is the number of considered attributes (metrics). We also assign a binary label r_c to each candidate c for each object v in validation set $\mathcal{V}$, indicating whether c is a relevant recommendation for v (r_c=1) or not (r_c=0), based on the contents of Y_v, the gold standard for tag recommendation. Note that we use training set $\mathcal{D}$ only to extract the association rules and to compute the metrics, relying on validation set $\mathcal{V}$ to learn the solutions (see discussion in Section 5.2). Thus, we only assign labels r_c for objects in $\mathcal{V}$. The learned model, i.e., the ranking function $f(M_c)$, is then applied to each candidate c for each object o in the test set. All reported results are based on the performance on the test sets.

Next, we describe the analyzed L2R techniques, and how they were applied to tag recommendation.

4.3.1 *RankSVM Based Strategy.* RankSVM is based on the state-of-the-art Support Vector Machine (SVM) classification method [Joachims 2006]. We use the SVM-rank tool[3] to learn a function $f(M_c){=}f(W, M_c)$, where $W = < w_1, w_2, ...w_m >$ is a vector of weights associated with the considered metrics (i.e., $W \in \mathbb{R}^m$). W is learned by a maximum-margin optimization method that tries to find a hyperplane, defined by W, that best separates the "closest" candidate tags (represented by their metric vectors in $\mathbb{R}^m$) belonging to two different *levels of relevance* (i.e., relevant and irrelevant) assigned to each object-candidate pair in the training. They are employed to produce ranking statements (i.e., relevant tags must precede irrelevant ones), which in turn are used as input to the RankSVM learning process. At recommendation time, $f(M_c)$ is used to rank all candidates for target object o according to their relative distances to the separating hyperplane. RankSVM has 2 key parameters: the type of kernel function, which indicates the structure of the solution function, and cost j, which controls the penalty to classification errors in the training process.

4.3.2 *Random Forest Based Strategy.* The Random Forest (RF) algorithm [Breiman 2001] is an ensemble method that combines a collection of decision trees. The learning of each decision tree in the ensemble happens in a recursive way: first, the most discriminative attribute (according to some measure, such as Information Gain) is selected as a decision node. The selected candidate tags are split according to a split value (e.g., average attribute value), and the process repeats in a top-down fashion to form a tree with l terminal nodes, where l is a tuning parameter. Once the decision tree is built, it can assign a real-valued score as output for an unseen (test) candidate tag.

The *RF* method exploits the *bagging* ensemble technique, i.e., each tree within the forest is built with a different bootstrap sample drawn from the original set of pairs (M_c, r_c) that represents each candidate tag c for the considered objects in our dataset, where, as discussed before, $M_c \in \mathbb{R}^m$ and $r_c \in \{0, 1\}$. The attribute selection for each split in a tree is conducted on a randomly selected subset of attributes, instead of on the full attribute set, as usually done in traditional decision tree algorithms. Each leaf in each tree corresponds to an output score to be assigned to a candidate tag. Once the forest is built, for each tag candidate c in a target object o, the scores given by each tree to c are averaged and used to produce the final ranking of candidates.

Besides l, the number T of trees to grow and the number of attributes m to consider when splitting each node are tuning parameters in RF. We note that, although each decision tree may suffer from overfitting, the aggregation of a larger number of low-correlated trees can mitigate this problem. The generalization error of a RF depends on both the correlation between trees in the forest and the strength of each individual tree. The more correlated each tree is, the higher the error rate becomes.

[3] `http://www.cs.cornell.edu/People/tj/svm_light/svm_rank.html`

The stronger each individual tree is (high accuracy), the lower the error rate becomes. By increasing m or lowering l, both the correlation and the strength of each tree increases. By lowering m or increasing l, each tree becomes more independent (less correlated), but also becomes weaker at the same time. Thus, there exists some optimal values of m and l that provide the optimal balance between the correlation and the strength to get the minimum generalization error. We set those parameters using the validation set, as described in Section 5.2. The implementation of RF and the next four approaches were provided by the RankLib learning to rank tool[4].

RF has been shown consistently effective and competitive in several real world benchmarks [Mohan et al. 2011]. Some of its strengths are its insensitivity to parameter choices, resilience to overfitting, and high degree of parallelization due the fact that single decision trees are built independently from others, thus making RFs inherently parallel.

4.3.3 *MART Based Strategy.* Similarly to RF, $MART$ (Multiple Additive Regression Trees) [Friedman 2000] combines multiple decision trees. However, unlike RF, $MART$ exploits the *boosting* ensemble technique, which is based on the idea that it is easier to find and average many weak learners, than to find a single, highly accurate predictor. Boosting is an iterative method that produces and combines different *weak learners*. In each of i iterations, it increases emphasis on training instances which were not well modeled in the previous iteration (i.e., candidate tags in the training set which were not correctly ranked by the model built in the previous iteration, in our case). Increasing the weight given to these "harder" training examples, it potentially improves the final model, which is a linear combination of the weak learners, with weights defined by the learning rate lr, a tuning parameter. Each weak learner is a decision tree with l leaves. However, unlike RF, $MART$ does not build randomized trees, i.e., it considers the whole training set and the complete set of attributes.

4.3.4 *λ-MART Based Strategy.* λ-MART [Wu et al. 2010], the winning approach at the Yahoo! Learning to Rank Challenge [Chapelle and Chang 2011], is a combination of MART and the λ-Rank ranking model, which tries to directly optimize the value of an evaluation metric. The main difference between λ-MART and MART is that λ-MART learns which candidate in a pair of candidate tags must appear first in the ranking. In order to learn these pairwise preferences, λ-MART uses the gain in Normalized Cumulative Discounted Gain (NDCG) [Baeza-Yates and Ribeiro-Neto 1999] from swapping the rank positions of candidates in any given pair of candidate tags for the same object o. Thus, unlike MART, the training is made by considering that each candidate tag is in the set of candidate tags C_o for an object o. Similarly to MART, the learning rate lr, the number of terminal nodes l and the number of iterations i must be specified.

4.3.5 *ListNet Based Strategy.* The goal of the learning in ListNet [Cao et al. 2007] is minimizing ranking errors, rather than minimizing errors in classification of pairs of tags or building regression models. ListNet is based on comparing the probability distribution of permutations of lists of candidate tags. Specifically, for a set of candidate tags to an object o, ListNet first defines a permutation probability distribution based on the scores produced by a ranking function for each candidate tag. It then defines another distribution based on the ground truth labels, and measures the inconsistency between these two distributions. The ranking function is defined as a neural network model. Thus, it is possible to iteratively adjust this ranking function according to the measured inconsistency. This adjustment is done at a pre-defined learning rate lr during i iterations.

4.3.6 *AdaRank Based Strategy.* The basic idea of AdaRank [Xu and Li 2007] is to plug a selected evaluation metric into the *boosting* framework and directly optimize this metric. Specifically, it repeatedly builds weak rankers on the basis of re-weighted training of sets of candidate tags, i.e., each set of candidate tags C_o for an object o receives a weight. The weak rankers are linearly combined

[4] `http://people.cs.umass.edu/~vdang/ranklib.html`

to make ranking predictions. Each attribute in Table I, in isolation, is used as a weak ranker. The selected weak ranker in each iteration of the algorithm corresponds to the attribute that leads to the best performance over the weighted objects, measured by a given evaluation metric (NDCG, in our case). AdaRank runs for i iterations. At each iteration, AdaRank maintains a distribution of weights over the objects in the training data. Initially, AdaRank sets equal weights to the objects. At each iteration, it increases the weights of those objects whose tags are not ranked well by the model created so far. As a result, the learning at the next iteration will be focused on the creation of a weak ranker that can work on the tag ranking of those "hard" objects.

4.3.7 *RankBoost Based Strategy.* Similarly to AdaBoost, RankBoost [Freund et al. 2003] also adopts the boosting framework, i.e., it trains one weak learner at each iteration, and learns a linear combination of weak rankers as the final ranking function. Each weak ranker consists of a single attribute (one of the metrics in Table I) and a threshold that best distinguishes between relevant and non-relevant candidate tags. However, similarly to RankSVM, RankBoost operates on pair of tags. After each iteration, the tag pairs are re-weighted: it decreases the weight of correctly ranked pairs and increases the weight of wrongly ranked pairs. As a result, the learning at the next iteration will be focused on dealing with pairs which are more difficult to rank. The total number of iterations i is a tuning parameter.

4.3.8 *GP Based Strategy.* Genetic Programming (GP) is a framework inspired by the biological mechanisms of genetic inheritance and evolution of individuals in a population [Banzhaf et al. 1998]. GP implements a global search mechanism by evolving a population of individuals over multiple generations. Each individual, representing a possible solution for the target problem (a tag ranking function), is modeled as a tree composed of terminals (leaves) and operators (inner nodes), related to the target problem. In our case, terminals are constants (uniformly distributed between 0 and 1) and attributes (metrics in Table I), while the inner nodes are operators sum, subtraction, multiplication as well as protected division and logarithm (so that they return the default value 0 if their inputs are out of their domains). In each generation, each individual is evaluated by a *fitness* function, defined based on quality metrics related to the problem at hand. We here use the NDCG metric as fitness function. Only individuals with the highest fitness values are selected, according to some selection method (we adopt the tournament selection, i.e., selecting the best individual among k randomly chosen individuals), to evolve the population.

An initial randomly generated population is evolved in a number of generations, through *crossover* and *mutation* operations. The crossover operation, performed on two selected individuals with probability p_c, is implemented by randomly choosing one node of each tree representing a selected individual and exchanging the subtrees below them. It aims at combining good solutions towards a more promising one. The mutation operation, on the other hand, adds new individuals (solutions) to the population, thus increasing the diversity in it. This is useful, for instance, to avoid being trapped in local optima. With probability p_m, the mutation of a selected individual is done by first randomly choosing one node of its tree, and then replacing the subtree rooted at it by a new randomly generated subtree, without exceeding a maximum tree depth d. Note that population size n_p is kept fixed through all generations. This process continues until a target fitness value f^t or a maximum number of generations n_g is reached. The individual with the best fitness value, usually part of the last generation, is chosen as the final solution for the problem.

GP is a non-linear method that has been applied to various IR tasks. We were the only to use it for recommending tags [Belém et al. 2011; Belém et al. 2013], having obtained competitive (or superior) results over RankSVM.

The main features of the above L2R techniques are summarized in Table II.

Table II. Characteristics of our L2R strategies.

Approach	Ensemble	Training instances	Directly optimize evaluation metric?	Generated model
RankSVM	-	Pairs of candidate tags	no	Hyperplane
RF	bagging	Sets of candidate tags	no	Set of "randomized" decision trees
MART	boosting	Sets of candidate tags	no	Set of boosted decision trees
λ-MART	boosting	Sets of candidate tags	yes	Set of boosted decision trees
ListNet	-	Sets of candidate tags	no	Neural network
AdaRank	boosting	Sets of candidate tags	yes	Sets of weighted weak rankers
RankBoost	boosting	Pairs of candidate tags	no	Sets of weighted weak rankers
GP	-	Sets of candidate tags	yes	Any function formed by a given set of operators and attributes

5. EXPERIMENTAL SETUP

In this section, we describe the datasets used to evaluate the tag recommendation strategies (Section 5.1), our evaluation methodology (Section 5.2), and how we parameterized each strategy (Section 5.3).

5.1 Data Collections

We evaluate the tag recommendation methods on five datasets, each containing the *title*, *tags* and *description* associated with real objects from Bibsonomy, LastFM, MovieLens, YouTube and YahooVideo. The Bibsonomy dataset is a recent snapshot of the system comprising 543,872 objects (bibtex records of publications), obtained on January 1^{st} 2012. It is publicly available[5], and has been used in several previous efforts [Lipczak and Milios 2011; Rendle and Schmidt-Thie 2010; Yin et al. 2013]. The MovieLens dataset, also available online[6], contains 10,000 objects. The LastFM and YouTube datasets were collected[7] in August 2009, following a *snowball sampling*. That is, starting from a set of users (the most popular users) selected as seeds, the crawler recursively collected the objects posted by them and followed their social links to other users, collecting the objects posted by them as well. Our datasets include the textual features associated with 2,758,992 LastFM artists and more than 9 million YouTube videos. The YahooVideo dataset was also gathered by snowball sampling, but using the most popular objects as seeds and following links of related videos. It was gathered in October 2008, and contains the features of 160,228 objects.

We considered only objects with textual features in English, removed stopwords, and used the Porter Stemming algorithm[8] to remove affixes of the words in the collected features. Stemming was performed to avoid trivial recommendations (e.g., plurals and simple variations of the same word).

5.2 Evaluation Methodology

Recall that our goal is to compare the use of alternative L2R techniques to recommend tags in terms of effectiveness, which relates to the quality of the recommended tags, and efficiency, measured by the time required to perform a recommendation. We focus on relevance as main criterion of quality, leaving other aspects such as novelty and diversity to future work [Belém et al. 2013].

As in the work of Rendle and Schmidt-Thie [2010], Menezes et al. [2010] and Yin et al. [2013], we adopt a fully-automated methodology to evaluate the effectiveness of these techniques: we use a subset of the tags previously assigned to the target object o as a *gold standard*, i.e., as the relevant tags for o. As such, these tags are disregarded for the computation of metrics of relevance. This methodology

[5] `http://www.kde.cs.uni-kassel.de/bibsonomy/dumps`.
[6] `http://www.grouplens.org/taxonomy/term/14`
[7] Visit `http://vod.dcc.ufmg.br/recc/` for information on data availability.
[8] `http://tartarus.org/~martin/PorterStemmer/`

was adopted because the manual evaluation of tag relevance is a very expensive process that may be affected by the subjectivity of human judgments. We note that the results obtained according to the proposed methodology represent lower bounds, as some of the recommended tags, although not in the gold standard, might still be considered relevant for the given object or user. Following this methodology, for each tuple $\langle I_o, F_o, Y_o \rangle$ in the test and validation sets, we randomly select half of its tags to be included in I_o. The other half is included in Y_o, the gold standard for o. We also use title and description as textual features in F_o, for each object o.

We sampled N tuples from each dataset, with N being 150,000 for YouTube, LastFM and Bibsonomy, 120,000 for YahooVideo, and 6,500 for MovieLens. Each sample was divided into five equal-sized portions, which were used in a five-fold cross validation. That is, three portions were treated as training set ($\mathcal{D}$) and used for extracting association rules and computing all metrics. A fourth portion was used as validation set $\mathcal{V}$, which, in turn, was considered a part of the training set, being used to "learn" the solutions and to tune parameters of all recommendation methods. The last portion was used for testing. Note that our use of the 5-fold cross-validation process for the L2R based strategies is slightly different from the traditional use: we learn the ranking function and select parameter values in the validation set $\mathcal{V}$. We do so to avoid overfitting, which could occur if the solutions were learned in the same set from which association rules and metrics were extracted (i.e., training set $\mathcal{D}$), as metrics derived from the rules could be over-inflated[9].

We argue that our experimental design is fair because: 1) we do not use any privileged information from the test set in which results of all methods are reported; 2) all parameters are discovered in the same validation set; 3) our collections are large enough for effective learning; and 4) the very tight confidence intervals reported in our results are evidence of low variation and thus learning convergence. Moreover, having a large amount of training data to generate the tag co-occurrence rules can help increasing the coverage of the rules (i.e., more co-occurrences can be potentially found), thus generating more candidates and more precise metric values. This benefits all methods.

Our primary metric to evaluate the effectiveness of each method is the Normalized Cumulative Discounted Gain (NDCG) [Baeza-Yates and Ribeiro-Neto 1999], computed over the ranked list of candidate tags produced for each object in the test set. We also measure the *precision*, i.e., the fraction of recommended tags that are relevant, and the *recall*, i.e., the fraction of the relevant tags for an object that are indeed recommended. All three metrics are computed over the top k positions of the ranking, with k=5.

In order to evaluate the efficiency of the methods, we measured their online recommendation time, which is an important performance measure since tag recommendation is an interactive task. Thus, the user may abandon the system if the waiting time is unacceptable. Model training, i.e., learning the recommendation functions and best parameters, can be performed completely offline, being thus less concerning. All experiments were performed on a 16-core 2.40GHz Intel(R) Xeon processor, with 50GB RAM. We measured CPU time (user and system time) for the batch of executions consisting of all objects in the test set. Reported results are averages per object, computed over all 5 test sets.

For all methods, recommendation time is divided into four stages: (1) lazy association rule generation, (2) metric computation, (3) application of the (learned) model to score candidate tags, which depends on the complexity of the generated model (e.g., number of nodes of decision trees), and (4) sorting of the candidate tags according to the scores. All stages can be somehow benefited by data storage in cache (e.g., pre-computed attribute values and association rules [Menezes et al. 2010], or even the final ranking of tags for a given object). However, we here focus on the worst case (an upper-limit of execution time), assuming that the cache will be always empty. Thus, all metrics and necessary association rules are computed at recommendation time. As such, the ranking (application of the recommendation function and sorting of candidate tags) will also be always performed. We

[9]The overfitting problem was observed in initial tests, with the learned solutions having lower generalization capabilities.

leave the evaluation of the benefits from alternative caching strategies for future work.

Table III. Best parameterization for each approach.

Approach	**Parameter**
LATRE+wTS	$\alpha = 0.95$ for LastFM and MovieLens, $\alpha = 0.9$ for Bibsonomy, YahooVideo and YouTube
RankSVM	linear kernel, $j = 100$
GP	$n = 200$, $k = 2$, $d = 7$, $pc = 0.6$ and $pm = 0.1$
RankBoost	$i = 500$ for Bibsonomy, $i = 700$ for LastFM, i=300 for MovieLens, YahooVideo and YouTube
RF	$T = 300$, $m = 4$, $l = 1000$ for Bibsonomy and LastFM, $l = 300$ for the remaining datasets
MART	$l = 5$, $lr = 0.1$ and $i = 1500$
λ-MART	$l = 5$, $lr = 0.1$ and $i = 1500$
AdaRank	$i = 300$ for Bibsonomy, LastFM, MovieLens and YahooVideo, $i = 100$ for YouTube
ListNet	$lr = 0.00001$, $i = 160$ for MovieLens, $i = 10$ for YahooVideo, $i = 40$ for Bibsonomy, LastFM and YouTube.

5.3 Parameterization

We ran a series of experiments with the validation set $\mathcal{V}$ to find the best parameters of each method. We found our *RF*-based tag recommender to be very insensitive to parameterization. The results obtained in our cross-validation experiments using different number of trees (T=300, 500, 1000) are statistically tied (with 95% confidence) for all datasets. We thus set T=300, due to the lower cost. We also fixed the number m of all attributes selected in each split of the tree according to the default value originally suggested by Breiman [2001], i.e $m = \lfloor log_2(M + 1) + 0,5 \rfloor$, where M is the total number of attributes. Despite the fact that this default value has been reported to work well in practice, we verified that other values ranging from $0.25M$ to $0.75M$ do not significantly impact our results. The only parameter that (slightly) impacts results is the number of terminal nodes l. We used cross validation to determine the best l among values from 10^2 to 10^3, finding that the best choice is l=1000 for Bibsonomy and LastFM, and $l = 300$ for MovieLens, YahooVideo and YouTube.

The number of leaves l also impacts *MART* and *λ-MART*, the other tree-based approaches. We experimented with l between 2 and 20, finding l=5 as the best choice for all datasets. Since the results obtained with different number of iterations ($i = 1500, 3000, 6000$) are statistically tied in all *MART* and *λ-MART* experiments, we set i=1500 in all experiments due the lower cost. We also varied learning rate lr between 0.0001 and 0.2, finding that the best choice was 0.1 for all datasets. Greater values for both l and lr do not improve effectiveness, and make these methods very inefficient.

For *ListNet*, our results were not very sensitive to the lr parameter. We tested values ranging from 10^{-7} to 10^{-1}, finding that $lr = 10^{-5}$ always led to the best results. We also varied the number of iterations i between 10 and 10^3, finding, as best choices, $i = 160$ for MovieLens, $i = 10$ for YahooVideo, and $i = 40$ for the other datasets. Similarly, we tested ten values of i between 10^2 and 10^3 for *RankBoost* and *AdaRank*. For *AdaRank*, $i = 300$ was the best choice in most datasets, except for YouTube, where $i = 100$ was the best value. For *RankBoost*, the best values varied according to the dataset: it was set to 500 in Bibsonomy, 700 in LastFM and 300 in the other datasets.

Regarding *LATRE+wTS*, *RankSVM* and *GP*, we adopted the same best parameter values reported by Belém et al. [2011] for LastFM, YouTube and YahooVideo, since the datasets are the same. For MovieLens and Bibsonomy, which were not included in that work, we follow the same methodology. Using cross-validation in $\mathcal{V}$, we found j=100 as the best cost for *RankSVM*, and we used the linear kernel. For *GP*, we set n=200 and $g = 200$ (as in the work by Belém et al. [2011]), $k = 2$, $d = 7$, $p_c = 0.6$ and $p_m = 0.1$, as usually done in the literature [Banzhaf et al. 1998]. Finally, for *LATRE+wTS*, the best parameter values are α=0.9 for Bibsonomy and $\alpha = 0.95$ for MovieLens. We set ℓ=3 for the metric *Sum* (for all methods), as in Belém et al. [2011].

We summarize our parameterization of all methods in Table III. Unless otherwise noted, the same parameter value was used for all datasets.

6. REPRESENTATIVE RESULTS

Now we report the results of experiments with objects in the *test sets*, comparing the results of all analyzed methods, i.e., the state-of-the-art unsupervised $LATRE{+}wTS$ heuristic and the eight L2R-based strategies RF, $MART$, λ-$MART$, $AdaRank$, $ListNet$, $RankSVM$, GP and $RankBoost$. Our comparison comprises both effectiveness (Section 6.1) and efficiency (Section 6.2) of the analyzed methods. Results are averages of all 5 folds, along with 95% confidence intervals.

6.1 Recommendation Effectiveness

Tables IV-VI show results for NDCG, precision and recall, respectively, considering the top k=5 ranked tags. We start by noting that the best L2R-based strategies (i.e., RF and λ-$MART$) outperform the state-of-the-art unsupervised $LATRE{+}wTS$ by up to 29%, 23% and 22% in NDCG, precision and recall, respectively. These gains are even more expressive than the gains achieved by previously evaluated L2R methods [Belém et al. 2011]. They confirm the benefits of exploiting supervised L2R methods for tag recommendation, allowing an automatic search for a solution that combines a larger number of attributes when compared to unsupervised heuristics such as $LATRE{+}wTS$.

We now turn our attention to the comparison of the eight L2R-based strategies. Unlike existing comparisons of different L2R techniques in other domains (e.g., document ranking) [Gomes et al. 2013], there is a clear winning group of methods (RF, $MART$ and λ-$MART$) in all 5 datasets, with a slight advantage of two of them (RF and λ-$MART$). The gains in NDCG of the winner methods over the best of the remaining L2R techniques considered (i.e., either GP, $RankSVM$ or $RankBoost$) range from 4% to 12%. The corresponding gains in precision and recall reach 10% and 11%, respectively. These results confirm the effectiveness of methods based on an ensemble of decision trees, which are non-linear L2R strategies that have been shown to be effective and competitive in other studies [Mohan et al. 2011; Breiman 2001].

The second group of methods is formed by the L2R strategies previously exploited in tag recommendation: GP, $RankSVM$ and $RankBoost$. GP is the most flexible strategy, allowing a wider range of types of recommendation functions (any function formed by the considered operators and attributes). However, this can be also a disadvantage because the search space is larger when compared to the search space of other methods, making it more difficult to find the best function. On the other hand, the shape of functions produced by $RankSVM$ is pre-defined by the kernel function, which was set linear here (as this led to the best results), and thus all $RankSVM$-produced functions consist of linear combinations of the attributes. Although $RankBoost$ is composed by simpler weak rankers (defined by single attribute), it achieved results similar to $RankSVM$, since its ensemble strategy also produces a linear combination of the attributes.

Comparing the general results across different datasets, we note that the best results are for YahooVideo, while MovieLens (which is considerably smaller than the other datasets) and Bibsonomy present the worst results. The observed differences are possibly due to the number of tags per object, which tends to be smaller in MovieLens and Bibsonomy than in YahooVideo objects. For example, 48% and 73% of our YahooVideo and YouTube objects have fewer than 10 tags, against 94%, 88% and 76% of the objects in Bibsonomy, LastFM, and MovieLens, respectively. Moreover, these results may also be due to differences in tagging behavior: on YahooVideo and YouTube, tags tend to appear in other textual features of the same object more often than on LastFM [Belém et al. 2011; Figueiredo et al. 2013] and the other datasets, which facilitate the recommendation of relevant tags exploiting some of the considered tag relevance metrics (e.g., wTS).

In sum, we found that (1) L2R based strategies can significantly outperform state-of-the-art unsupervised heuristics, and (2) RF, λ-$MART$ and $MART$ are the best L2R strategies out of the eight analyzed techniques, providing further gains over previously evaluated L2R-based strategies.

Table IV. Average NDCG@5 values and 95% confidence intervals. Best results (and statistical ties) in bold.

	Bibsonomy	LastFM	MovieLens	YahooVideo	YouTube
LATRE+wTS	0.397± 0.001	0.388 ± 0.003	0.321 ± 0.009	0.744 ± 0.002	0.489 ± 0.001
RankSVM	0.412 ± 0.002	0.407 ± 0.002	0.354 ± 0.007	0.765 ± 0.001	0.515 ± 0.003
GP	0.406 ± 0.006	0.440 ± 0.008	0.388 ± 0.002	0.770 ± 0.0044	0.530 ± 0.002
RankBoost	0.444 ± 0.002	0.402 ± 0.002	0.307 ± 0.005	0.696 ± 0.003	0.488 ± 0.002
RF	**0.495 ± 0.003**	**0.469 ± 0.002**	**0.413 ± 0.004**	**0.803 ± 0.002**	**0.549 ± 0.002**
MART	0.489 ± 0.002	0.463 ± 0.001	0.411 ± 0.006	0.794 ± 0.002	0.547 ± 0.001
λ-MART	**0.493 ± 0.002**	**0.468 ± 0.002**	**0.409 ± 0.010**	**0.802 ± 0.003**	**0.551 ± 0.002**
AdaRank	0.446 ± 0.004	0.160 ± 0.128	0.206 ± 0.110	0.653 ± 0.010	0.419 ± 0.032
ListNet	0.431 ± 0.006	0.380 ± 0.007	0.273 ± 0.012	0.607 ± 0.003	0.472 ± 0.003

Table V. Average Precision@5 values and 95% confidence intervals. Best results (and statistical ties) in bold.

	Bibsonomy	LastFM	MovieLens	YahooVideo	YouTube
LATRE+wTS	0.438 ± 0.002	0.401 ± 0.002	0.314 ± 0.008	0.733 ± 0.003	0.489 ± 0.002
RankSVM	0.456 ± 0.003	0.419 ± 0.002	0.346 ± 0.006	0.754 ± 0.002	0.517 ± 0.003
GP	0.441 ± 0.009	0.450 ± 0.006	0.363 ± 0.004	0.755 ± 0.005	0.520 ± 0.002
RankBoost	0.451 ± 0.003	0.424 ± 0.002	0.366 ± 0.002	0.763 ± 0.003	0.517 ± 0.002
RF	**0.502 ± 0.003**	**0.494 ± 0.001**	**0.386 ± 0.006**	**0.797 ± 0.002**	**0.543 ± 0.002**
MART	0.496 ± 0.002	0.489 ± 0.001	0.385 ± 0.002	0.792 ± 0.002	0.541 ± 0.001
λ-MART	**0.501 ± 0.002**	**0.493 ± 0.002**	**0.385 ± 0.003**	**0.797 ± 0.002**	**0.546 ± 0.001**
AdaRank	0.454 ± 0.003	0.134 ± 0.063	0.180 ± 0.149	0.712 ± 0.010	0.440 ± 0.038
ListNet	0.437 ± 0.006	0.398 ± 0.008	0.316 ± 0.010	0.661 ± 0.003	0.499 ± 0.003

Table VI. Average Recall@5 values and 95% confidence intervals. Best results (and statistical ties) in bold.

	Bibsonomy	LastFM	MovieLens	YahooVideo	YouTube
LATRE+wTS	0.430 ± 0.002	0.377± 0.002	0.250 ± 0.012	0.637± 0.002	0.451 ± 0.001
RankSVM	0.446 ± 0.003	0.390 ± 0.003	0.275 ± 0.008	0.651 ± 0.001	0.474 ± 0.003
GP	0.431 ± 0.009	0.415 ± 0.010	0.280 ± 0.005	0.654 ± 0.004	0.478 ± 0.002
RankBoost	0.441 ± 0.002	0.394 ± 0.002	0.287 ± 0.007	0.657 ± 0.003	0.474 ± 0.002
RF	**0.491 ± 0.003**	**0.460 ± 0.002**	**0.301 ± 0.011**	**0.689 ± 0.001**	**0.498 ± 0.002**
MART	0.485 ± 0.002	0.455 ± 0.002	0.300 ± 0.007	0.685 ± 0.002	0.496 ± 0.001
λ-MART	**0.490 ± 0.002**	**0.459 ± 0.002**	**0.298 ± 0.011**	**0.690 ± 0.002**	**0.502 ± 0.001**
AdaRank	0.443 ± 0.004	0.152 ± 0.120	0.194 ± 0.105	0.618 ± 0.009	0.408 ± 0.030
ListNet	0.428 ± 0.006	0.374 ± 0.006	0.259 ± 0.012	0.575 ± 0.003	0.459 ± 0.003

6.2 Recommendation Efficiency

Now we show results for the efficiency of all analyzed methods in terms of their online recommendation time. Table VII shows the average recommendation time per object, in *milliseconds*, broken down into each of the 4 recommendation stages (see Section 5.2). In particular, we distinguish the metric computation (MC) stage for the L2R techniques, which involve the computation of all metrics used[10], and the MC stage for $LATRE{+}wTS$, which consists of the computation of only wTS. Table VIII shows the total recommendation time (sum of all stages) per object, on average, in *seconds*.

As expected, the simplest unsupervised method ($LATRE{+}wTS$) takes less time than the others, since it requires only the computation and sum of two metrics – Sum and wTS – besides the on-demand association rule generation and candidate sorting, which are common steps to all methods. Note however that, in terms of total recommendation time, the use of several of the analyzed L2R strategies only incur in a small additional cost (under 3%) in comparison to $LATRE{+}wTS$.

Comparing the recommendation time of the L2R strategies, we found that, despite some variation, the most efficient models are those generated by *RankBoost*-based strategies, followed by $RankSVM$, $ListNet$ and GP. The three winner methods in terms of effectiveness come next. In any case, the average recommendation time of all methods, in a worst case scenario (without any pre-computed metric or association rule), is under 1.3 seconds per object.

[10]The exception is $Sum(\ell = 3)$, whose computation time is included in the lazy rule generation (LRG) step, for all (supervised and unsupervised) methods analyzed.

Table VII. Average recommendation time (milliseconds) per object, divided in stages: Lazy Rule Generation (LRG), Metric Computation (MC), Model Application (MA) and Candidate Sorting (CS)

		Bibsonomy	LastFM	MovieLens	YahooVideo	YouTube
LRG		116.06 ± 2.46	1198.11 ± 72.97	238.00 ± 12.10	968.18 ± 84.77	592.13 ± 100.71
MC (L2R methods)		0.53 ± 0.01	0.96 ± 0.04	0.49 ± 0.02	0.58 ± 0.01	0.97 ± 0.02
MC ($LATRE$+wTS)		0.04 ± 0.01	0.10 ± 0.01	0.03 ± 0.02	0.04 ± 0.00	0.03 ± 0.01
MA	$LATRE$+wTS	0.02 ± 0.01	0.04 ± 0.01	0.02 ± 0.01	0.02 ± 0.01	0.03 ± 0.01
	$RankSVM$	0.72 ± 0.04	1.52 ± 0.07	0.46 ± 0.00	0.65 ± 0.08	1.35 ± 0.06
	GP	2.12 ± 0.13	5.35 ± 0.53	2.23 ± 0.23	1.95 ± 0.12	4.99 ± 0.56
	$RankBoost$	0.59 ± 0.02	1.30 ± 0.02	1.38 ± 0.08	0.53 ± 0.02	1.17 ± 0.09
	RF	47.20 ± 1.66	54.43 ± 0.31	799.23 ± 45.00	55.63 ± 1.73	57.18 ± 1.41
	$MART$	10.24 ± 0.12	19.11 ± 0.19	82.38 ± 0.62	10.23 ± 0.18	16.01 ± 0.69
	λ-$MART$	50.76 ± 9.83	60.69 ± 32.13	472.00 ± 119.92	52.29 ± 9.66	90.02 ± 35.64
	$AdaRank$	1.48 ± 0.23	4.53 ± 0.69	3.92 ± 0.85	1.33 ± 0.09	4.11 ± 1.27
	$ListNet$	1.43 ± 0.24	2.79 ± 1.17	2.38 ± 1.08	1.05 ± 0.19	2.91 ± 0.34
CS		0.04 ± 0.01	0.08 ± 0.01	0.05 ± 0.01	0.03 ± 0.01	0.07 ± 0.01

Table VIII. Average recommendation time (seconds) per object

	Bibsonomy	LastFM	MovieLens	YahooVideo	YouTube
$LATRE$+wTS	0.116 ± 0.002	1.198 ± 0.073	0.238 ± 0.012	0.968 ± 0.085	0.592 ± 0.101
$RankSVM$	0.117 ± 0.003	1.201 ± 0.073	0.239 ± 0.012	0.969 ± 0.085	0.595 ± 0.101
GP	0.119 ± 0.003	1.205 ± 0.074	0.241 ± 0.012	0.971 ± 0.085	0.598 ± 0.101
$RankBoost$	0.117 ± 0.002	1.200 ± 0.073	0.240 ± 0.012	0.969 ± 0.085	0.594 ± 0.101
RF	0.164 ± 0.004	1.254 ± 0.073	1.038 ± 0.057	1.024 ± 0.087	0.650 ± 0.102
$MART$	0.127 ± 0.003	1.218 ± 0.073	0.321 ± 0.013	0.979 ± 0.085	0.609 ± 0.101
λ-$MART$	0.167 ± 0.012	1.260 ± 0.105	0.711 ± 0.132	1.021 ± 0.094	0.683 ± 0.136
$AdaRank$	0.118 ± 0.003	1.204 ± 0.074	0.242 ± 0.013	0.970 ± 0.085	0.597 ± 0.102
$ListNet$	0.118 ± 0.003	1.202 ± 0.074	0.241 ± 0.013	0.970 ± 0.085	0.596 ± 0.101

In conclusion, the most cost-effective solution is the $MART$-based method, as it produces results very close to RF and λ-$MART$ (e.g., the difference is under 1.3% in NDCG), with a much shorter recommendation time (differences range from 185% to 473%). Thus, despite some variation across different methods, the L2R approach is feasible for the tag recommendation task. Moreover, L2R methods are flexible, allowing the inclusion of new attributes and addressing other problem aspects.

7. CONCLUSIONS AND FUTURE WORK

We have presented a comparative study of nine tag recommendation strategies: one state-of-the-art unsupervised heuristic ($LATRE$+wTS) and eight supervised learning-to-rank (L2R) based methods, with respect to both effectiveness (i.e., precision, recall and NDCG) and efficiency (i.e., time complexity). Our experiments showed that the best analyzed L2R-based strategy outperforms the state-of-the-art heuristic, producing gains of up to 29% in NDCG. Among the L2R based strategies, there is a clear winner group of methods: Random Forests (RF), $MART$ and λ-$MART$, which produces gains ranging from 4% to 12% in NDCG over the best of the remaining L2R methods (i.e., the previously proposed methods GP, $RankSVM$ and $RankBoost$). In terms of efficiency (in recommendation time) the use of L2R does not incur in an expressive additional cost with relation to the unsupervised alternative. Despite some variation across the L2R methods, we found that they are feasible for the tag recommendation task: in a worst-case scenario, the average recommendation time per object is under 1.3 seconds. Moreover, L2R represent a flexible approach, allowing the inclusion of new attributes and other aspects of the problem (e.g., personalization) which we plan to address in the future, besides performing a user study to evaluate the recommendation effectiveness.

REFERENCES

Baeza-Yates, R. and Ribeiro-Neto, B. *Modern Information Retrieval.* Addison-Wesley, Boston, MA, USA, 1999.

Banzhaf, W., Francone, F., Keller, R., and Nordin, P. *Genetic Programming: an introduction on the automatic evolution of computer programs and its applications.* Morgan Kaufmann Publishers, San Francisco, USA, 1998.

Belém, F., Martins, E., Almeida, J., and Gonçalves, M. Exploiting Novelty and Diversity in Tag Recommendation. In *Proceedings of ECIR on Advances in Information Retrieval*. Moscow, Russia, pp. 380–391, 2013.

Belém, F., Martins, E., Pontes, T., Almeida, J., and Gonçalves, M. Associative Tag Recommendation Exploiting Multiple Textual Features. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. Beijing, China, pp. 1033–1042, 2011.

Breiman, L. Random forests. *Machine Learning* 45 (1): 5–32, 2001.

Cao, H., Xie, M., Xue, L., Liu, C., Teng, F., and Huang, Y. Social Tag Prediction Base on Supervised Ranking Model. In *Proceedings of the ECML PKDD Discovery Challenge*. Bled, Slovenia, pp. 35–48, 2009.

Cao, Z., Qin, T., Liu, T., Tsai, M., and Li, H. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proceedings of the International Conference on Machine Learning*. Corvalis, Oregon, pp. 129–136, 2007.

Chapelle, O. and Chang, Y. Yahoo! Learning to Rank Challenge Overview. *Journal of Machine Learning Research* vol. 14, pp. 1–24, 2011.

Faria, F., Veloso, A., Almeida, H., Valle, E., Torres, R., Gonçalves, M., and Meira, W. Learning to Rank for Content-Based Image Retrieval. In *Proceedings of the International Conference on Multimedia Information Retrieval*. Philadelphia, Pennsylvania, USA, pp. 285–294, 2010.

Figueiredo, F., Belém, F., Pinto, H., Almeida, J., and Gonçalves, M. Assessing the Quality of Textual Features in Social Media. *Information Processing & Management* 49 (1): 222–247, 2013.

Freund, Y., Iyer, R., Schapire, R., and Singer, Y. An Efficient Boosting Algorithm For Combining Preferences. *Journal Of Machine Learning Research* vol. 4, pp. 933–969, 2003.

Friedman, J. Greedy Function Approximation: a gradient boosting machine. *Annals of Statistics* vol. 29, pp. 1189–1232, 2000.

Gomes, G., Oliveira, V., de Almeida, J., and Gonçalves, M. Is Learning to Rank Worth It? A Statistical Analysis of Learning to Rank Methods. *Journal of Information and Data Management* 4 (1): 57–66, 2013.

Heymann, P., Ramage, D., and Garcia-Molina, H. Social Tag Prediction. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. Singapore, Singapore, pp. 531–538, 2008.

Joachims, T. Training Linear SVMs In Linear Time. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Philadelphia, PA, USA, pp. 217–226, 2006.

Lipczak, M. and Milios, E. Efficient Tag Recommendation for Real-Life Data. *ACM Transactions on Intelligent Systems and Technology* 3 (1): 2:1–2:21, 2011.

Menezes, G., Almeida, J., Belém, F., Gonçalves, M., Lacerda, A., Moura, E., Pappa, G., Veloso, A., and Ziviani, N. Demand-Driven Tag Recommendation. In *Proceedings of the International Conference on Information and Knowledge Management*. Barcelona, Spain, pp. 402–417, 2010.

Mohan, A., Chen, Z., and Weinberger, K. Web-Search Ranking with Initialized Gradient Boosted Regression Trees. *Journal of Machine Learning Research, Workshop and Conference Proceedings* vol. 14, pp. 77–89, 2011.

Pedro, J. S., Siersdorfer, S., and Sanderson, M. Content Redundancy in YouTube and its Application to Video Tagging. *ACM Transactions on Information Systems* vol. 29, pp. 13:1–13:31, 2011.

Rendle, S. and Schmidt-Thie, L. Pairwise Interaction Tensor Factorization for Personalized Tag Recommendation. In *Proceedings of the ACM Conference on Web Search and Data Mining*. New York, USA, pp. 81–90, 2010.

Sigurbjornsson, B. and van Zwol, R. Flickr Tag Recommendation based on Collective Knowledge. In *Proceedings of the International Conference on World Wide Web*. Beijing, China, pp. 327–336, 2008.

Song, Y., Zhang, L., and Giles, C. Automatic Tag Recommendation Algorithms for Social Recommender Systems. *ACM Transactions on the Web* vol. 5, pp. 1–31, 2011.

Wang, J., Hong, L., and Davison, B. D. Tag Recommendation Using Keywords And Association Rules. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Bled, Slovenia, 2009.

Wu, L., Yang, L., Yu, N., and Hua, X. Learning To Tag. In *Proceedings of the International Conference on World Wide Web*. Madrid, Spain, pp. 361–370, 2009.

Wu, Q., Burges, C., Svore, K., and Gao, J. Adapting Boosting for Information Retrieval Measures. *Information Retrieval* 13 (3): 254–270, 2010.

Xu, J. and Li, H. AdaRank: a boosting algorithm for information retrieval. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. Amsterdam, Netherlands, pp. 391–398, 2007.

Yin, D., Guo, S., Chidlovskii, B., D.Davison, B., Archambeau, C., and Bouchard, G. Connecting Comments and Tags: improved modeling of social tagging systems. In *Proceedings of the ACM Conference on Web Search and Data Mining*. Rome, Italy, pp. 547–556, 2013.